Arab J Sci Eng (2014) 39:5279–5288
DOI 10.1007/s13369-014-1162-z

RESEARCH ARTICLE - SYSTEMS ENGINEERING

Solving the Multi-Mode Resource Availability Cost Problem in Project Scheduling Based on Modified Particle Swarm Optimization

Jian-Jun Qi · Ya-Jie Liu · Hong-Tao Lei · Bo Guo

Received: 18 December 2012 / Accepted: 15 July 2013 / Published online: 31 May 2014

Abstract This paper proposes a model for multi-mode resource availability cost problem (MMRACP) in project scheduling which minimizes the resource availability cost required to finish all activities in a project at a given project deadline. Precedence relations exist among the activities of the project in the model. Furthermore, renewable and nonrenewable resources are both considered. MMRACP is a nondeterministic polynomial time hard (NP-hard) problem, as a result it is very difficult to use an exact method to solve it. For solving MMRACP, we developed a modified particle swarm optimization method combined with path relinking procedure and designed a heuristic algorithm to improve the fitness of the solution. At the end, a computational experiment including 180 instances was designed to test the performance of the modified particle swarm optimization. Comparative computational results show that the modified particle swarm optimization is very effective in solving MMRACP.

Keywords Project scheduling · MMRACP · Path relinking · Particle swarm optimization · NP-hard · Heuristics

J.-J. Qi (✉) · Y.-J. Liu · H.-T. Lei · B. Guo
College of Information System and Information Management, National University of Defense Technology, Changsha, 410073 Hunan, Republic of China
e-mail: hustqjj@126.com

B. Guo
e-mail: boguo@nudt.edu.cn

الخلاصة

تعرض هذه الورقة العلمية مسألة تكلفة توفر الموارد متعددة النمط بوصفها نموذجاً في جدولة المشروع التي تقلل من تكلفة توفر الموارد اللازمة لإنهاء كل النشاطات في مشروع خلال الوقت المعين المعطى للمشروع. ويحتوي هذا النموذج على علاقات الأسبقية بين أنشطة المشروع. وتؤخذ بعين الاعتبار -إضافة إلى ذلك-الموارد المتجددة وغير المتجددة. إن مسألة تكلفة توفر الموارد متعددة الأنماط هي مسألة ذات حدود متعددة ووقت صلب وغير حتمية، ونتيجة لذلك فإنه من الصعب جداً استخدام طريقة معينة لحلها. ولذلك قمنا بتطوير طريقة من سرب الجسيمات الأمثل مركبة مع طريقة إعادة ربط المسار وقمنا بتصميم خوارزمية إرشادية بهدف تحسين ملاءمة الحل. وفي النهاية تم تصميم تجارب حاسوبية تتضمن 180 حالة لاختبار أداء طريقة سرب الجسيمات الأمثل التي تم تعديلها. وتظهر النتائج الحسابية أن طريقة سرب الجسيمات الأمثل التي تم تعديلها فعالة جداً في حل مسألة تكلفة توفر الموارد متعددة الأنماط.

1 Introduction

Profitability is the most important factor for a contractor, on the condition that the project is accomplished within the specified deadline. When the project deadline is confirmed, contractors usually attempt to minimize the total cost of the project. Thus, job schedules and resource availability are very important for the contractor. This leads to the resource availability cost problem (RACP).

The earliest study on RACP was by Möhring [1] motivated by a bridge construction project, and an exact algorithm based on graph theory was proposed in his study. Yamashita [2, 3] adopted a scatter search mechanism to solve RACP and created robust model of RACP. Shadrokh [4] proposed a genetic algorithm for RACP to minimize penalty. Ranjbar [5] solved the problem by a path relinking and genetic algorithm. Rodrigues [6] developed an exact algorithm that consists of a hybrid method in which the initial feasible solution is found heuristically. These studies have good performance in solving RACP.

Actually, most jobs in a project have more than one mode for selection and the duration of the job is changed with the mode. For example, the job of bricklaying needs four master bricklayers to work for three days or three master bricklayers to work for four days. This is a very simple case in a project, wherein the proper scheme is selected according to the situation. Therefore, the multi-mode RACP (MMRACP) is more prevalent than the single-mode RACP in practice. The most familiar problem with MMRACP is the multi-mode resource-constrained project scheduling problem (MMRCPSP) [7–13], which is also a nondeterministic polynomial time hard (NP-hard) problem that involves job scheduling, mode selection of the job, and amounts of renewable and nonrenewable resources. MMRCPSP is so complicated that determining the best scheduling is very difficult when the number of jobs exceeds 30. Most researchers try to use artificial intelligence algorithms, to solve this problem such as evolution algorithm [8], genetic algorithm [9], estimation of distribution algorithm (EDA) [12] and so on. Most of these studies target on minimizing the duration of the project. However, the resource availability cost problem concerned by the contractors has not yet been solved well.

Particle swarm optimization (PSO) is a widely used artificial intelligence algorithms proposed by Kennedy [14], which can be applied extensively in solving tough problems [15–17]. Biswas solved the machine-loading problems in flexible manufacturing systems based on pseudo PSO. Deng [15] proposed a pseudo elite archiving method adopting the PSO algorithm to solve the injection moulding optimization problem. Chen [17] used PSO with justification and designed mechanisms for RCPSP. From these articles, PSO was known to have a very good performance in solving scheduling problems. In this study, MMRACP is investigated. The model of MMRACP is created with the following assumptions: (1) deadline of the project is confirmed that has no relationship with the other factors; (2) total cost of the project is the mainly issue that concerned by the contractor, aside from the project's deadline; (3) renewable and nonrenewable resources are both required in the project; and (4) there exist precedence relationships among different jobs. MMRACP is more complicated than RACP and MMRCPSP. Thus, a modified particle swarm optimization (MPSO) combined with path relinking procedure is proposed to solve MMRACP.

The remaining content of this article is organized as follows. In Sect. 2, the model of MMRACP is proposed. MPSO for solving MMRACP is presented in the third section. The computational experiment and computational results are presented in Sect. 4. Conclusions are made in the last section.

2 Problem Descriptions

RACP can be stated as follows [1–5]. A project composed of $n + 2$ activities, including n real activities and two dummy activities. Each activity has its own duration and requires some units of renewable resource over its duration. Finish–start-type precedence relations exist among the activities, and the deadline of the project is confirmed. The objective of RACP is to find the best feasible schedule for minimizing total cost of the project.

In MMRACP, most activities have multiple candidate modes for selection and the durations are changed depending on the selected mode. During the duration of the activity, renewable and nonrenewable resources are both required. MMRACP has no differences with RACP on the other aspects such as precedence relations, deadline and so on.

2.1 Mathematical Model

This section provides the mathematical formulation of MMRACP. The notation used is shown as follows:

N :	Number of activities;
i :	Index of the activity;
N_i :	Mode number of the ith activity;
m :	Mode index of the activity;
x_{im} :	Whether the mth mode is selected for activity i. If selected, $x_{im} = 1$, otherwise, $x_{im} = 0$;
K_r :	Number of renewable resource types;
K_n :	Number of nonrenewable resource types;
K :	Number of resource types and $K = K_r + K_n$;
$k(k = 1, 2, \ldots K)$:	Index of resource types. If $k \leq K_r$, the kth resource is renewable resource, else nonrenewable resource;
a_k :	Amount of the kth resource type that the project needs.
p_k :	Whether the kth resource type is nonrenewable, if nonrenewable $p_k = 1$; otherwise, $p_k = 0$;
r_{imk} :	Amount of resource type k that needed by the ith activity in the mth mode;
d_{im} :	Duration required to finish the ith activity as needed in the mth mode;
s_i :	Start time of the ith activity;
D :	Deadline of the project;
C_k :	Cost of one unit resource of type k;
t :	The tth time unit;
A_t :	Set of activities in progress during the time interval $(t - 1, t]$

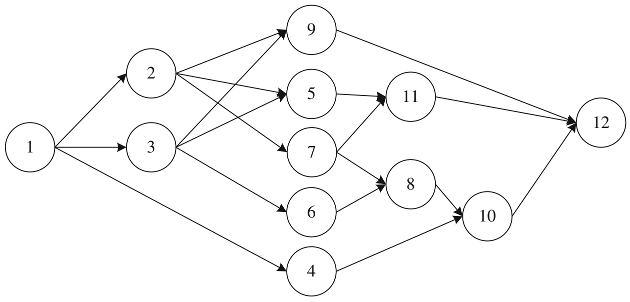


Fig. 1 Precedence relations of the activities

The objective of the MMRACP is to minimize total cost of the project, which is equivalent to the sum of resource cost, including both renewable and nonrenewable resource.

$$\text{Minimize} \quad f = \sum_{k\in R} C_k a_k \tag{1}$$

The constraints are defined as follows:

$$\sum_{m=1}^{M_i} x_{im} = 1, i = 1, 2, \ldots N_i \tag{2}$$

$$s_i + \sum_{m=1}^{M_i} d_{im} x_{im} \leq s_j \tag{3}$$

$$s_{N+2} \leq D \tag{4}$$

$$\sum_{i=1}^{i=N+2} \sum_{m=1}^{N_i} x_{im} r_{imk} p_k = p_k a_k \quad \forall k \in R \tag{5}$$

$$\sum_{i\in A_t} \sum_{m=1}^{N_i} r_{imk} |p_k - 1| \leq a_k \tag{6}$$

The constraint denoted in Eq. (2) indicates that each activity has only one mode for selection. Equation (3) shows that the schedules must satisfy the finish-start-type precedence relation among the activities, where activity j is a successor of activity i and $\sum_{m=1}^{M_i} d_{im} x_{im}$ is the duration of activity i. Equation (4) ensures that the project is finished before the deadline. Equation (5) determines that the sum of the nonrenewable resource required for each activity equal to $p_k a_k$. Equation (6) requires that the sum of the renewable resource used at each time unit in progress should be no more than a_k.

2.2 An Example

In this section, an instance is provided. A project consists of 12 activities including two dummy activities and ten real activities, and the precedence relations are shown in Fig. 1.

Table 1 Detailed information of the activities

Jobnr.	Mode	Duration	R1	R2	N1	N2
1	1	0	0	0	0	0
2	1	9	0	5	0	4
2	2	10	2	0	0	4
2	3	10	2	0	10	0
3	1	2	0	3	7	0
3	2	3	0	3	5	0
3	3	7	10	0	5	0
4	1	7	6	0	0	8
4	2	7	0	2	10	0
4	3	8	6	0	0	7
5	1	6	7	0	9	0
5	2	6	0	5	8	0
5	3	10	0	3	6	0
6	1	1	5	0	0	2
6	2	3	4	0	10	0
6	3	10	3	0	6	0
7	1	4	10	0	10	0
7	2	5	0	2	0	7
7	3	5	0	2	10	0
8	1	4	0	7	0	8
8	2	5	5	0	9	0
8	3	5	5	0	0	7
9	1	3	4	0	8	0
9	2	6	3	0	0	6
9	3	9	2	0	3	0
10	1	4	5	0	0	6
10	2	5	0	4	2	0
10	3	6	5	0	0	5
11	1	7	7	0	0	2
11	2	8	0	7	0	2
11	3	9	0	3	5	0
12	1	0	0	0	0	0

Each real activity has three modes for selection. Furthermore, two types of renewable resource (R1 and R2) and two types of nonrenewable resource (N1 and N2) are required in the project. Detailed information of the activities is shown in Table 1. Assuming that the deadline is 30 days and one unit cost of each type resource is given, the problem is to find a feasible schedule minimizing resource availability cost.

As MMRACP is a NP-hard problem, using an exact method to solve this problem is difficult. For the instance shown in Table 1 and Fig. 1, more than $3^{10} \times (10!) \approx 2.1 \times 10^{11}$ schedules exist, including infeasible and feasible schedules. Thus, the best method available is the use of intelligent algorithm to find the "best" solution.

Table 2 Two schedules for the project

Number activity	Schedule 1		Schedule 2	
	Start time	Mode	Start time	Mode
2	1	1	0	1
3	9	1	7	1
4	2	1	7	3
5	13	1	9	3
6	11	1	16	1
7	12	2	12	2
8	17	1	17	1
9	11	3	17	3
10	21	2	21	2
11	19	1	19	1

Two schedules are shown in Table 2. Schedule 1 needs 9 units of resource R1 ($a_1 = 9$), 10 units of R2 ($a_2 = 10$), 18 units of N1 ($a_3 = 18$), and 31 units of N2 ($a_4 = 31$). However, $a_1 = 9$, $a_2 = 10$, $a_3 = 16$, and $a_4 = 30$ in schedule 2. Thus, schedule 2 is better than schedule 1.

3 Modified Particle Swarm Optimization for Solving MMRACP

In this section, the principle of PSO is introduced in the first part. Then, the schedule representation for solving MMRACP is shown in the second part. And the third part represents a heuristic algorithm for repairing the particles and decoding scheme. Path relinking procedure combined with PSO is proposed in the fourth section. At the end, the flowchart of modified PSO is given.

3.1 Principle of PSO

PSO-incorporated swarming behaviours are observed in flocks of birds, schools of fish, swarms of bees, and even human social behaviour [18,19]. It consists of a swarm of particles in a space, wherein the position vector of each particle presents a solution. At the beginning, the particles are randomly positioned, looking for the best position with best fitness. In each generation, each particle moves to a new position as a new solution, which is guided by a velocity. The velocity is influenced by the position of the global and local best experiences.

Let N denote the dimension of the space and M represent the number of particles in the space. N is typically concerned with the definition of the problem. Let $X_i = [X_{i1}, X_{i2}, \ldots, X_{iN}]$represent the position of the particle i. The velocity of the particle i is $V_i = [V_{i1}, V_{i2}, \ldots, V_{iN}]$, and the local best experience of the individual particle is $L_i = [L_{i1}, L_{i2}, \ldots, L_{iN}]$. $G = [G_1, G_2, \ldots, G_N]$ represents the global best experience of all particles. The component j of particle i is updated according to the following equation:

$$\begin{cases} V_{ij}(t+1) = \omega V_{ij}(t) + c_1 r_1 (L_{ij}(t) - V_{i,j}(t)) \\ \qquad\qquad +c_2 r_2 (G_{ij}(t) - V_{ij}(t)) \\ X_{ij}(t+1) = X_{ij}(t) + V_{ij}(t+1) \end{cases} \tag{7}$$

where ω is the inertia weight used to determine the influence of the previous velocity on the new velocity. c_1 and c_2 are the learning factors that drive the particle in approaching a new position, maintaining the proper distance in an individual local and global best experience. In addition, r_1 and r_2 are the random numbers uniformly distributed in [0, 1], controlling the trade-off between the global and local exploration abilities during the search.

Normally, PSO promotes cooperation instead of rivalry, and saves the beneficial information of the particles, while most of other meta-heuristics discard them [20]. As a result, the convergence of PSO is very fast. PSO is widely used to combine with other algorithms, such as simulated annealing (SA) [21–23], DNA computing [24] and so on [25].

3.2 Solution Representation and Initial Population

For MMRACP, the position vector of a solution is represented by two subvectors: the scheduling vector containing the scheduling information and the mode vector devoted to the mode information.

For the scheduling information, the most popular representation used in solving the scheduling problem is a serial schedule generation scheme. However, it is not a good choice for solving RACP because of the indeterminate amount of resource. Thus, the start time is used as the component of the position vector in this study. Following this way, the objective function could be calculated directly. Furthermore, the schedule can avoid the influence of the uncertainty of resource. The shortcoming of this schedule representation is that most of the schedules are not feasible without considering the precedence determined in Sect. 3.3.

For activities with different modes, the shorter the duration is, the larger the mode number is and the more resources needed are. The mode number of the activity is observed to express a great deal of useful information. Thus, the mode number can be used to represent the mode vector of the solution, which is suitable for PSO.

With the above description, a multi-mode encoding scheme is used to transform the representation to a schedule. An example is shown as follows:

$S = (s_1, s_2, \ldots s_i, \ldots s_N, m_1, m_2, \ldots m_i, \ldots m_N)$

in which $s_i (0 \leq s_i < D)$ denotes the start time of the ith activity and $m_i (m_i = 1, 2, \ldots N_i)$ indicates that the m_ith mode is selected for the ith activity.

A good solution representation is crucial for solving the NP-hard problem, besides, the method of creating the initial population is very important. Most researchers propose heuristics to solve scheduling problems efficiently [26,27]. The initial solutions are built step-by-step as follows:

Step 1: The mode information is randomly generated;

Step 2: The earliest start time EST_j and latest finish time LFT_j of activity j are calculated based on the information of the mode, precedence relations and deadline of the project.

Step 3: The position vectors of particle i are randomly assigned as listed in Eq. (8):

$$X_{ij} = \lfloor \mathrm{EST}_j + \mathrm{ram} \times (\mathrm{LFT}_j - \mathrm{EST}_j) \rfloor \tag{8}$$

where ram is a random number uniformly distributed in [0, 1] and $X_{ij} = \lfloor a \rfloor$ is the biggest integer smaller than a.

3.3 Heuristics to Repair the Particles and Decoding Scheme

The appearance of the infeasible solutions during the process of optimization is inevitable. Thus, repairing the particle is necessary when decoding the scheme if the solution of the particle is infeasible. The heuristic to repair the particles is shown as follows:

Step 1: Convert the position vector of each particle to be a schedule. As the position vector of each particle is composed of real numbers, it is necessary to convert the real numbers to be integer numbers to generate a schedule. For the scheduling information, the transformation process is shown as follows:

$$s_i = \lfloor X_i \rfloor \quad i = 1, 2, \ldots N \tag{9}$$

For the mode information, the transformation process is as listed in Eq. (10):

$$m_i = \lfloor X_{i+N} \rfloor \quad i = 1, 2, \ldots N \tag{10}$$

As the mode vector is not involved in the process of repairing the solution after this step, it is necessary to make sure that the mode vector does not lead to the appearance of infeasible solution. If $m_i > N_i$, let $m_i = N_i$. If $m_i < 1$, let $m_i = 1$.

Step 2: Repair the particle to make the solution meet the precedence relations among the activities. All activities are checked from the first to the last. If the successor of activity i starts before activity i finished, the start time of the successor is set equal to the finish time of activity i. If activity j is a successor of activity i, it must be $j > i$. This step continues further until that all activities meet the precedence restrictions. The pseudo-code is shown in algorithm 1.

Algorithm.1 Pseudo-code of step 2

```
1: int mark=0;
2: for each activity from $i=1$ to $N$ do
3:     for each activity $j$ from $j=i+1$ to $N$ do
4:         if ( $f_i > s_j$ )and (activity $j$ is a successor
           of activity $i$ ) then
5:             $s_j = f_i$;
6:             bfeasible=false;
7:         end if
8:     end for
9: end for
10: Go to step 3;
```

Step 3: Let all activities start as early as possible and make sure whether the duration of this solution overrun the deadline of the project. If the solution is infeasible, go to step 7. Otherwise, go to step 4. The pseudo-code of this step is shown in algorithm 2, in which EA denotes the start time of the foremost start activity and LA represents the end time of the last finish activity. EA is subtracted from the start time of all activities to reduce the duration of the whole project. If $LA - EA \leq D$, it means that the project is finished before the deadline. Otherwise, the solution is infeasible.

Algorithm.2 Pseudo-code of step 3

```
1:  $EA = D$;
2:  $LA = 0$ ;
3:  for each activity from $i=1$ to $N$ do
4:      if $EA > s_i$ then
5:          $EA = s_i$;
6:      end if
7:      if $LA < f_i$ then
8:          $LA = f_i$;
9:      end if
10: end for
11: for each activity from $i=1$ to $N$ do
12:     $s_i = s_i - EA$;
13: end for
14: if $LA - EA > D$ then
15:     Go to Step 4;
16: else
17:     Go to step 7;
18: end if
```

Step 4: Sort all activities in an ascending order by the start time. For each activity, the position in the array and the value of the start time are both needed in step 4. As a result, its position and value are both saved during the process of sorting activities. For the ith start activity, the value of $sort\_act_i$ is the index of the activity and the value of $sort\_time_i$ is the start time of the $sort_act_i$th activity.

Algorithm.3 Pseudo-code of step 4

```
1: for each activity from $i=1$ to $N$ do
2:     $sort\_act_i = i$;
3:     $sort\_time_i = s_i$;
4: end for
5: for each $sort\_time_i$ from $i=1$ to $N$ do
6:     for each $sort\_time_j$ from $j=i+1$ to $N$ do
7:         if $sort\_time_i > sort\_time_j$ then
8:             Exchange values between $sort\_time_i$
               and $sort\_time_j$;
9:             Exchange values between $sort\_act_i$ and
               $sort\_act_j$;
10:        end if
11:    end for
12: end for
13: Go to step 5;
```

Step 5: Repair the particle to make the duration of the project shorter. The pseudo-code of this step is shown in algorithm 4. As a result of that the first starting activity must start at time 0, the activities are repaired from $i = 2$ to n. For activity $sort\_act_i$, all activities are visited from $j = 1$ to $j = i - 1$. If the $sort\_act_j^{\text{th}}$ activity has a resource competition with activity $sort\_act_i$ and is finished before the start of activity $s_{sort\_act_i}$, or activity $sort\_act_j$ is a predecessor of activity $sort\_act_i$, compare $\max\_start$ with $f_{sort\_act_j}$: If $\max\_start < f_{sort\_act_j}$, $\max\_start = f_{sort\_act_j}$. Finally, let $s_{sort\_act_i} = \max\_start$.

Algorithm.4 Pseudo-code of step 5

```
1: for each $sort\_act_i$ from $i=2$ to $N$ do
2:     Int $max\ start = 0$;
3:     for each $sort\_act_j$ from $j=1$ to $i$ do
4:         if ( $sort\_act_j$ is $sort\_act_i$'s predecessor )
   or ( $s_{sort\_act_i} > f_{sort\_act_j}$ and resource competition exists
   between activity $sort\_act_i$ and $sort\_act_j$ ) then
5:             if $\max\_start < f_{sort\_act_j}$ then
6:                 $\max\_start = f_{sort\_act_j}$;
7:             end if
8:         end if
9:     end for
10:    $s_{sort\_act_i} = \max\_start$;
11: end for
12: Go to step 6;
```

Step 6: Repair the particle last time to make sure that the solution is feasible. All activities are checked from the first to the last. If the start time of the checked activity is larger than its latest start time, the start time is equal to its latest start time. Then, go to step 7.

Step 7: End

Although it is complicated to repair the schedules, it is very easy to calculate the objective function following Eqs. (5) and (6) shown in Sect. 2.1.

3.4 Path Relinking Procedure

The path relinking procedure is used to generate new solutions from existing solutions. The best solution of the first generation is regarded as the initial solution (x_1). After this, PSO is used to find a better solution (*global*), which is regarded as the guide solution (x_2). The relinking method is then used to generate new solutions: if a better solution is not found, $x_1 = x_2$ and PSO is used to find a new solution; otherwise the best experienced solution is updated. Then let $x_1 = global$ and PSO is continued until a new better solution is found. The pseudo-code of the path relinking procedure is shown in algorithm 5.

Algorithm.5 Pseudo-code of path relinking procedure

```
1: $X_1$ =the best experienced position;
2: if better solution (global) is found based on PSO then
3:     $X_2$ =global;
4:     for each $i$ from 1 to $N$ do
5:         if $X_1[i] = X_2[i]$ then
6:             Let $i$ be the crossing points and $X_1$ and
               $X_2$ are father and mother solutions New
               solution1 and New solution2 are generated;
7:             $G_1$ =Decoding(New solution1 );
8:             $G_2$ =Decoding(New solution2 );
9:             if gbest value> $G_1$ then
10:                gbest value= $G_1$;
11:                global= New solution1 ;
12:            end if
13:            if gbest value> $G_2$ then
14:                gbest value = $G_2$;
15:                global= New solution2 ;
16:            end if
17:        end if
18:    end for
19:    $X_1$=global;
20: end if
```

In Algorithm 5, *New*_solution1 and *New_solution2* are the new solutions and Decoding (*New_solution*) is the function in calculating the fitness of *New_solution*. *gbest_value* denotes the fitness of the global best experience of all particles.

A solution is represented by two vectors, the position vector and the mode vector; thus, the two-point crossover is used to generate a new solution. The operator works as follows:

let $i (0 < i \leq N)$ be the first crossing point and $i + N$ be the second crossing point. Assume that *parent1* and *parent2* activity lists are given as follows:

$$parent1 = (p_1, p_2, \ldots, p_N, m_1, m_2, \ldots, m_N)$$

and

$$parent2 = (\alpha_1, \alpha_2, \ldots, \alpha_N, \beta_1, \beta_2, \ldots, \beta_N)$$

Then, two new solutions are generated as follows:

$$New\_solution1 = (p_1, \ldots, p_{i-1}, \alpha_i, \ldots, \alpha_N, m_1, \ldots, m_{i-1}, \beta_i, \ldots, \beta_N)$$

and

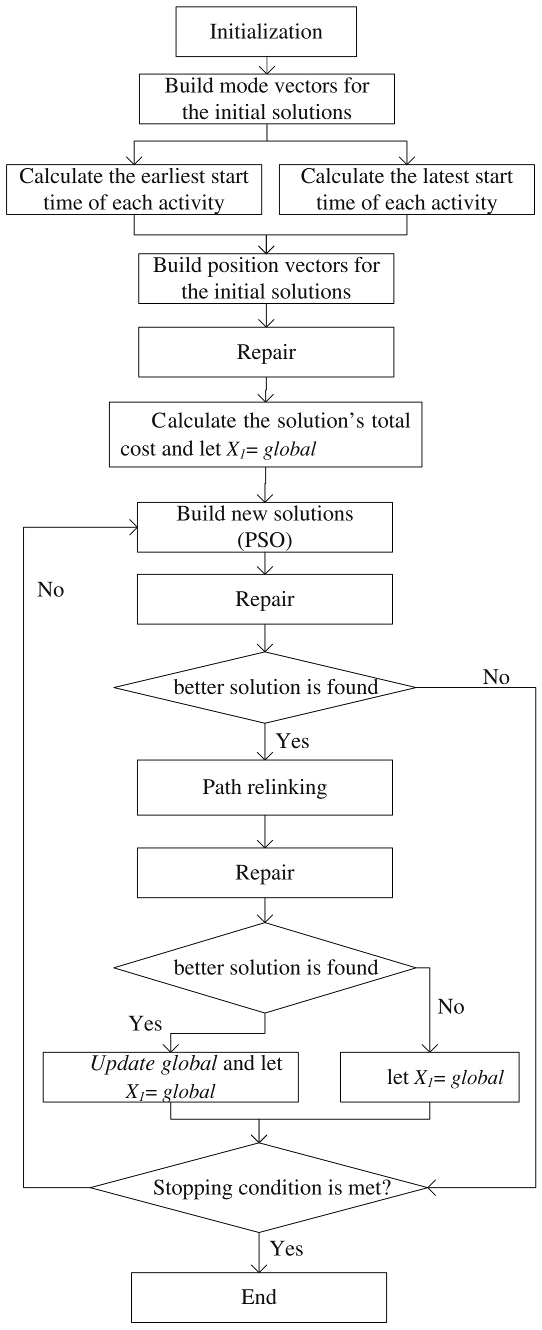
$$New\_solution2 = (\alpha_1, \ldots, \alpha_{i-1}, p_i, \ldots, p_N, \beta_1, \ldots, \beta_{i-1}, m_i, \ldots, m_N)$$

The advantage of the two-point crossover is that, for the same activity, both the position vector and the mode vector are changed simultaneously, which is more effective in generating solutions with quality.

3.5 Framework of the Modified PSO

With the above design, the procedure of the MPSO for solving the MMRACP is summarized in Fig. 2.

First, the initialization is implemented, and then the mode vectors for the initial solutions are built. The earliest start time and latest start time of each activity can be calculated based on the information of the mode, deadline of the project and precedence relations, which will be used to build the position vectors for the initial solutions according to the method shown in Sect. 3.2. After that, the schedules are repaired following the procedure shown in Sect. 3.3. The fitness of each solution is determined and the global experience solution can be found. Moreover, PSO is used to build new solutions. For each generation, if a better solution is found, the path relinking procedure is used to build new solutions. If a better solution is found during the path relinking procedure, the global experience swarm of the PSO is updated. No matter a better solution is found or not in the path relinking procedure, let "X_1=*global*". The PSO is then continued until that the stop condition is met. Finally, the "best" schedule is achieved. Straightforwardly, the framework of the modified PSO is illustrated in Fig. 2.



Initialization
Build mode vectors for the initial solutions
Calculate the earliest start time of each activity
Calculate the latest start time of each activity
Build position vectors for the initial solutions
Repair
Calculate the solution's total cost and let X_1= *global*
Build new solutions (PSO)
No
Repair
better solution is found
No
Yes
Path relinking
Repair
better solution is found
Yes
No
Update global and let X_1= *global*
let X_1= *global*
Stopping condition is met?
Yes
End

Fig. 2 Flowchart of the MPSO

4 Experimental Results

In this section, computational experiments were designed to test the performance of the modified PSO for MMRACP. The experiments were performed on a PC G630, 2.7 and 2.69 GHz with 2.99 GB of RAM. All procedures were coded in C language.

4.1 Generation of the Problem Instances

MMRACP and MMRCPSP share common data, therefore it is easy to adapt existing instances of the MMRCPSP to the MMRACP. In this study, the instances used are directly obtained from project scheduling problem library [28] (PSPLIB) involving 10, 12, 16, 18, 20, and 30 activities. Each instance contains three modes, two renewable and two non-renewable resources. There are some important parameters for the instances shown as follows:

- Resource factor (RF): reflects the density of different resource types needed by an activity and takes the values 0.5, 0.75 and 1. We did not distinguish renewable resource from nonrenewable resource when fixing RF. For the instance shown in section, RF is 0.5 which means that 50 % of the resource types are required by each real activity.
- Deadline factor (DF): reflects the deadline of the project, such that $D = \mathrm{DF} \times \max \mathrm{EF}_i$, where EF_i is the earliest finishing time of activity i on the condition that all activities choose the mode with the shortest duration. DF is fixed at 1.2.
- Object function: this function is considered as $f = \sum_{k \in R} C_k a_k$ where $C_k$are drawn from a uniform distribution $U[1, 10]$.

Seven important parameters for the modified PSO are as follows:

1. $\omega = 1$: an inertia weight used to determine the influence of the previous velocity to the new velocity;
2. $c_1 = 1$: the first learning factor;
3. $c_2 = 1$: the second learning factor;
4. $v_{\max\_\mathrm{P}} = 5$: max velocity of the position vector;
5. $v_{\min\_\mathrm{P}} = -5$: min velocity of the position vector;
6. $v_{\max\_\mathrm{M}} = 1$: max velocity of the mode vector;
7. $v_{\min\_\mathrm{M}} = -1$: min velocity of the mode vector.

For comparison, two important parameters [8] are used to evaluate the solution quality that can be obtained via the deviation from the best solution. According the studied problems, it is very difficult for us to get the best solution. Thus, MPSO, PSO, and genetic algorithm (GA) with 100,000 schedules, are used to find the almost "best" solution for each instance, named "up bound" solution. Each algorithm is executed five times and the best solution found is seemed as "up bound" solution. In this way, the deviation from the "up bound" solution is used to evaluate the quality of the solution, which is expressed as:

$$\mathrm{DEV}_i = \frac{\mathrm{fitness}_i - \mathrm{UB}_i}{\mathrm{UB}_i} \times 100\ \% \tag{11}$$

$$\mathrm{ADEV} = \frac{\sum_{i=1}^{N_{\mathrm{instaces}}} \mathrm{DEV}_i}{N_{\mathrm{instaces}}} \tag{12}$$

where N_{instaces} is the number of instances, UB_i is the fitness of the "up_bound" solution for the i^{th} instance and $\mathrm{fitness}_i$ denotes the fitness of the solution found by the used algorithm. DEV is the deviation from the "*up bound*"' solution and ADEV is the average deviation.

Table 3 The max settings of DEV in MPSO, PSO, and GA (50,000 schedules)

Activities	DF	RF	MPSO	PSO	GA
10	1.2	0.5	3.26	2.17	2.82
10	1.2	0.75	3.31	2.06	3.51
10	1.2	1	2.15	0.5	3.67
12	1.2	0.5	1.85	4.54	5.55
12	1.2	0.75	2.01	3.97	4.89
12	1.2	1	0.21	2.95	1.97
16	1.2	0.5	1.82	1.84	2.01
16	1.2	0.75	1.80	1.93	2.06
16	1.2	1	0.18	0.90	0.38
18	1.2	0.5	2.95	3.71	11.4
18	1.2	0.75	1.84	2.46	4.08
18	1.2	1	0.15	0.29	3.03
20	1.2	0.5	1.44	1.32	3.76
20	1.2	0.75	1.34	1.38	2.63
20	1.2	1	0.23	0.47	2.50
30	1.2	0.5	0.20	0.15	0.53
30	1.2	0.75	0.37	1.24	4.29
30	1.2	1	0.47	1.17	3.33
Global			1.2	1.84	3.47

4.2 Computational Experiment

In this section, computational experiments were designed. To the best of our knowledge, there were no researchers studying MMRACP before, so we had to design computational experiments by ourselves. The computational results from the implementation of MPSO, PSO, and GA have the same encoding and repairing methods.

Ten instances were included in each experiment group, wherein $6 \times 3 \times 10 = 180$ instances were used in the computational experiment. The settings of max DEV in MPSO, PSO, and GA with 50,000 schedules are shown in Table 3. The settings of ADEV in MPSO, PSO, and GA with 1,000, 10,000, and 50,000 schedules are shown in Table 4.

The following observations were obtained from the results:

1. If the instance is simple and the number of schedules is large enough, the performances of MPSO, PSO, and GA are similar;
2. If the instance is complicated and the number of schedules is not large enough, the performances of MPSO and PSO are much better than GA;
3. The performance of MPSO is almost the same with PSO when the schedules are not excessive, but it is more efficient than PSO as the size of the computation increases;
4. The performance of MPSO is steadier than PSO and GA;

Table 4 The settings of ADEV in MPSO, PSO, and GA

Activities	DF	RF	MPSO			PSO			GA		
			1,000	10,000	50,000	1,000	10,000	50,000	1,000	10,000	50,000
10	1.2	0.5	2.46	2.44	1.72	3.07	1.89	1.75	4.86	3.00	1.65
10	1.2	0.75	3.35	2.05	0.80	3.85	2.10	0.97	7.25	5.70	3.25
10	1.2	1	1.18	0.57	0.165	1.28	1.00	0.72	6.49	2.87	2.1
12	1.2	0.5	1.47	0.97	0.62	2.04	1.63	1.51	6.84	5.56	3.58
12	1.2	0.75	3.19	2.32	0.28	3.65	2.25	1.02	3.92	3.87	4.27
12	1.2	1	0.52	0.17	0.07	1.46	1.34	1.21	3.13	2.26	0.97
16	1.2	0.5	1.32	0.81	0.72	1.24	0.78	0.73	4.83	2.78	0.78
16	1.2	0.75	0.75	0.63	0.30	1.33	0.72	0.48	3.76	2.59	0.84
16	1.2	1	0.25	0.15	0.12	0.74	0.21	0.20	3.40	1.97	0.30
18	1.2	0.5	0.69	0.14	0.10	1.47	1.25	1.21	13.60	9.21	7.56
18	1.2	0.75	1.55	0.82	0.15	1.32	1.09	0.86	8.65	6.87	3.51
18	1.2	1	0.58	0.33	0.05	0.40	0.21	0.09	5.72	3.99	2.21
20	1.2	0.5	0.71	0.66	0.44	0.58	0.52	0.48	7.82	6.21	3.41
20	1.2	0.75	0.97	0.74	0.34	0.39	0.27	0.21	6.59	5.43	2.91
20	1.2	1	0.43	0.09	0.07	0.29	0.15	0.15	5.39	4.02	2.14
30	1.2	0.5	1.02	0.26	0.13	1.31	0.77	0.52	7.31	5.74	3.01
30	1.2	0.75	0.93	0.71	0.51	1.48	0.81	0.62	7.38	5.29	3.07
30	1.2	1	0.42	0.23	0.15	1.52	0.63	0.48	6.57	4.67	3.12
Global			1.21	0.78	0.37	1.52	0.98	0.73	6.31	4.56	2.70

Table 5 Average CPU-time (s) of the MPSO, PSO, and GA

	30	20	18	16	12	10
MPSO	2.10	0.9	0.87	0.46	0.42	0.28
PSO	2.06	0.88	0.85	0.42	0.4	0.27
GA	3.08	1.31	1.17	0.89	0.56	0.42

5. The performance of MPSO is more applied than that of PSO and GA in solving MMRACP problems.

The average CPU-time (s) of the MPSO, PSO, and GA with 50,000 schedules is shown in Table 5. The performance of MPSO is better than GA and slightly worse than PSO.

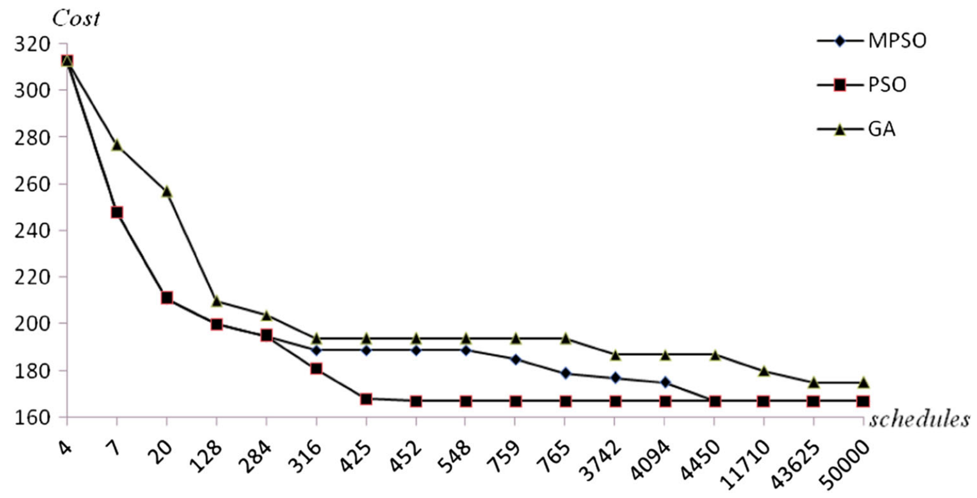


Fig. 3 Process of optimizing the instance in Sect. 2.2

As the ADEV and max DEV of MPSO are much better than PSO, the computational experiments prove that MPSO is the best choice for solving MMRACP. The process of optimizing the instance in Sect. 2.1 is shown in Fig. 3.

5 Conclusions

In this study, the model of MMRACP is established, which is much more applicable than RACP in practice. A modified PSO is presented to solve MMRACP, and a heuristic algorithm is designed to improve the fitness of the solution. All these procedures were tested on an available test. The computational results show that the modified PSO inherited the advantages of PSO and GA, which is more efficient in solving such complicated problem. The proposed method can provide high-quality solutions for large instances of MMRACP in a short time.

For further research, we recommend the fuzzy resource availability cost problem. The durations of most activities in the project are fuzzy in practice, thus the existed models about RACP have some limitations in solving the actual project scheduling problem. It is valuable to build the model of the fuzzy resource availability cost problem and provide an efficient heuristic algorithm to solve this problem.

Acknowledgments The authors sincerely thank Dr. Bassam El Ali and the anonymous referees. This research is partially supported by National Science Foundation of China (Grant no. 71201181 and 71201170).

References

1. Möhring, R.H.: Minimizing costs of resource requirements in project networks subject to a fix completion time. Oper. Res. **32**(1), 89–120 (1984)
2. Yamashita, D.S.; Armentano, V.A.; Laguna, M.: Scatter search for project scheduling with resource availability cost. Eur. J. Oper. Res. **169**(2), 623–637 (2006)
3. Yamashita, D.S.; Armentano, V.A.; Laguna, M.: Robust optimization models for project scheduling with resource availability cost. J. Sched. **10**(1), 67–76 (2007)
4. Shadrokh, S.; Kianfar, F.: A genetic algorithm for resource investment project scheduling problem, tardiness permitted with penalty. Eur. J. Oper. Res. **181**(1), 86–101 (2007)
5. Ranjbar, M.; Kianfar, F.; Shadrokh, S.: Solving the resource availability cost problem in project scheduling by path relinking and genetic algorithm. Appl. Math. Comput. **196**(2), 879–888 (2008)
6. Rodrigues, S.B.; Yamashita, D.S.: An exact algorithm for minimizing resource availability costs in project scheduling. Eur. J. Oper. Res. **206**(3), 562–568 (2010)
7. Sprecher, A.; Hartmann, S.; Drexl A.: An exact algorithm for project scheduling with multiple modes. OR Spektrum **19**(3), 195–203 (1997)
8. Damak, N.; Jarboui, B.; Siarry, P.; Loukil, T.: Differential evolution for solving multi-mode resource-constrained project scheduling problems. Comput. Oper. Res. **36**(9), 2653–2659 (2009)
9. Barrios, A.; Ballestín, F.; Valls, V.: A double genetic algorithm for the MRCPSP/max. Comput. Oper. Res. **38**(1), 22–43 (2011)
10. Deblaere, F.; Demeulemeester, E.; Herroel, W.: Reactive scheduling in the multi-mode RCPSP. Comput. Oper. Res. **38**(1), 63–74 (2011)
11. Wang, L.; Fang, C.: An effective shuffled frog-leaping algorithm for multi-mode resource- constrained project scheduling problem. Inf. Sci. **181**(20), 4804–4822 (2011)
12. Wang, L.; Fang, C.: An effective estimation of distribution algorithm for the multi-mode resource-constrained project scheduling problem. Comput. Oper. Res. **39**(2), 449–460 (2012)
13. Speranza, M.G.; Vercellis, C.: Hierarchical models for multi-project planning and scheduling. Eur. J. Oper. Res. **64**(2), 312–325 (1993)
14. Kennedy, J.; Eberhart, R.C.: Particle swarm optimization. In: Proceedings of IEEE International Conference on Nature Networks, pp. 1942–1948 (1995)
15. Deng, Y.M.; Zheng, D.; Lu, X.J.: Injection moulding optimisation of multi-class design variables using a PSO algorithm. Int. J. Adv. Manuf. Technol. **39**(7–8), 690–698 (2008)
16. Biswas, S.; Mahapatra, S.S.: Modified particle swarm optimization for solving machine-loading problems in flexible manufacturing systems. Int. J. Adv. Manuf. Technol. **39**(9–10), 931–942 (2008)
17. Chen, R.M.: Particle swarm optimization with justification and designed mechanisms for resource-constrained project scheduling problem. Expert Syst. Appl. **38**(6), 7102–7111 (2011)
18. Clerc, M.; Kennedy, J.: The particle swarm-explosion, stability, and convergence in a multidimensional complex space. IEEE Trans. Evol. Comput. **6**(1), 58–73 (2002)
19. Parsopoulos, K.E.; Vrahatis, M.N.: On the computation of all global minimizers through particle swarm optimization. IEEE Trans. Evol. Comput. **8**(3), 211–224 (2004)
20. Orosz, J.E.; Jacobson, S.H.: Analysis of static simulated annealing algorithms. J. Optim. Theory Appl. **115**(1), 165–182 (2002)
21. Triki, E.; Collette, Y.; Siarry, P.: A theoretical study on the behavior of simulated annealing leading to a new cooling schedule. Eur. J. Oper. Res. **166**(1), 77–92 (2005)
22. Shukla, S.K.; Son, Y.J.; Tiwari, M.K.: Fuzzy-based adaptive sample-sort simulated annealing for resource-constrained project scheduling. Int. J. Adv. Manuf. Technol. **36**(9–10), 982–995 (2008)
23. Khorasani, J.: A new heuristic approach for unit commitment problem using particle swarm optimization. Arab. J. Sci. Eng. **37**(4), 1033–1042 (2012)
24. Maghsoudi, M.J.; Ibrahim, Z.; Buyamin, S.; Rahmat, M.F.A.: Data clustering for the DNA computing readout method implemented on lightcycler and based on particle swarm optimization. Arab. J. Sci. Eng. **37**(3), 697–707 (2012)
25. Chen, S.P.; Vargas, Y.N.: Improving the performance of particle swarms through dimension reductions—a case study with locust swarms. In: 2010 IEEE Congress on Evolutionary Computation. IEEE Congress on Evolutionary Computation, pp. 1–8 (2010)
26. Debels, D.; De Reyck, B.; Leus, R.; Vanhoucke, M.: A hybrid scatter search/electromagnetism meta-heuristic for project scheduling. Eur. J. Oper. Res. **169**(2), 638–653 (2006)
27. Tormos, P.; Lova, A.: An efficient multi-pass heuristic for project scheduling with constrained resources. Int. J. Prod. Res. **41**(5), 1071–1086 (2003)
28. Project Scheduling Problem Library-PSPLIB. http://www.om-db.wi.tum.de/psplib/